Установка

Установка JumpServer Enteprise Edition

Оффлайн установка JumpServer:

Для начала нужно запросить у нас актуальный файл дистрибутива по почте support @afi-d.ru или в тг: @mapceaheh

1. Системные требования:
ОС: Linux/AMD64 (arm64) x86_64(aarch64) ядро версии 4.0 или выше (желательно семейства Redhat, Debian, Ubuntu)
CPU: 4 ядра
RAM: 8 ГБ
HDD: 60 ГБ

2. Поместить скачанный файл в директорию

```
/opt
```

3. Выполнить команды(имена файлов могут отличаться с выходом новых версий):

```
$ cd /opt
$ tar -xf jumpserver-offline-installer-v3.10.3-amd64.tar.gz
cd jumpserver-offline-installer-v3.10.3-amd64
```

Далее вы можете отредактировать файл конфигурации, для изменения параметров установки, например для использования внешней СУБД MySQL или изменению папки установки продукта.

nano /opt/jumpserver-offline-installer-v3.10.3-amd64/config-example.txt

Запускаем установку:

```
# cd jumpserver-offline-installer-v3.10.3-amd64
# ./jmsctl.sh install
```

Во время установки вам нужно будет подтвердить введенные в файле конфигурации данные или указать другие данные, если вы не заполняли файл конфигурации заранее.

Если вы планируете использовать внешний сервер СУБД, создайте БД по инструкции (https://t.me/JumpServerPAM/57).

Запускаем приложение:

```
# ./jmsctl.sh start
```

После завершения установки
перейти в папку с продуктом (имя папки может меняться с выходом новых версий)

```
# cd /opt/jumpserver-installer-v3.10.3
```

запустить приложение

```
# ./jmsctl.sh start
```

4. После этого можно будет зайти в веб-интерфейс по адресу:

http://IP/
логин: **admin**
пароль: **ChangeMe**

и приступить к настройке системы.

Установка JumpServer Community Edition

Внимание: *Если вы планируете пилотировать/использовать Enterprise редакцию с расширенными функциями, то нужно сразу ставить ее по этой инструкции (Установка JumpServer Enterprise Edition) , так как переход из Community в Enterprise невозможен без переустановки.*

1. Системные требования:
ОС: Linux/AMD64 (arm64) x86_64(aarch64) ядро версии 4.0 или выше
CPU: 4 ядра
RAM: 8 ГБ
HDD: 60 ГБ

2. Установка дополнительных компонентов на примере Debian\Ubuntu:

```
# apt-get update
# apt-get install -y wget curl tar gettext iptables
```

3. Установка JumpServer

Быстрая онлайн установка JumpServer:

В этом случае JumpServer установится с параметрами по-умолчанию, СУБД MySQL и Redis будут установлены в контейнеры на том же сервере.

Выполняем команду:

```
# curl -sSL https://github.com/jumpserver/jumpserver/releases/latest/download/quick_start.sh | bash
```

Дожидаемся процесса выполнения скрипта.

Стандартная онлайн установка:

В этом случае вы можете изменить конфигурацию установки перед развертыванием.

Скачайте последний установщик с GitHub https://github.com/jumpserver/installer/releases/
Ниже пример команд для версии 3.10.3

```
# cd /opt/
# wget https://github.com/jumpserver/installer/releases/download/v3.10.3/jumpserver-installer-v3.10.3.tar.gz
# tar -xf jumpserver-installer-v3.10.3.tar.gz
```

Далее вы можете отредактировать файл конфигурации, для изменения параметров установки.

Если вы планируете использовать внешний сервер СУБД, создайте БД по инструкции (https://t.me/JumpServerPAM/57).

```
# nano /opt/jumpserver-installer-v3.10.3/config-example.txt
```

Запускаем установку:

```
# cd ./jumpserver-installer-v3.10.3
# ./jmsctl.sh install
```

Во время установки вам нужно будет подтвердить введенные в файле конфигурации данные или указать другие данные, если вы не заполняли файл конфигурации заранее.

4. Запуск приложения

После завершения установки нужно перейти в папку с продуктом и запустить приложение

```
# cd /opt/jumpserver-installer-v3.10.3
# ./jmsctl.sh start
```

После этого можно будет зайти в веб-интерфейс по адресу:

http://IP/
логин: **admin**
пароль: **ChangeMe**
И приступить к настройке системы.

Эксплуатация и обслуживание с помошью командной строки - jmsctl

Эксплуатация и обслуживание - jmsctl

JumpServer по умолчанию включает встроенный инструмент командной строки для эксплуатации и обслуживания - **jmsctl.** Для просмотра справочных документов выполните команду:

```
jmsctl help
```

Управление приложением JumpServer:

```
./jmsctl.sh [COMMAND] [ARGS...]
./jmsctl.sh --help
```

Команды установки:

- install - Установка сервиса JumpServer

Команды управления:

- config - Конфигурация инструмента, выполните jmsctl config --help для просмотра справки
- start - Запуск сервиса JumpServer
- stop - Остановка сервиса JumpServer
- restart - Перезапуск сервиса JumpServer
- status - Проверка состояния сервиса JumpServer
- down - Остановка сервиса JumpServer
- uninstall - Деинсталляция сервиса JumpServer

Дополнительные команды:

- load_image - Загрузка Docker-образа
- backup_db - Резервное копирование базы данных JumpServer
- restore_db [file] - Восстановление данных из резервного файла базы данных
- raw - Выполнение команды docker compose
- tail [service] - Просмотр логов сервиса

Описание сетевых портов

Список сетевых портов

JumpServer требует открытия следующих сетевых портов для нормальной работы. Администраторы могут открыть соответствующие порты в сети и на хосте в зависимости от схемы развертывания компонентов JumpServer.

Порт	Назначение	Описание
22	SSH	Установка, обновление и управление
80	Web HTTP сервис	Доступ к веб-интерфейсу JumpServer по HTTP
443	Web HTTPS сервис	Доступ к веб-интерфейсу JumpServer по HTTPS
3306	Сервис базы данных	Используется MySQL
6379	Сервис базы данных	Используется Redis
3389	Razor сервисный порт	Подключение к Windows-активам через RDP Client
2222	SSH Client	Подключение к JumpServer через терминальные инструменты (Xshell, PuTTY и т.д.)
33061	Magnus MySQL сервисный порт	Подключение к MySQL через DB Client
33062	Magnus MariaDB сервисный порт	Подключение к MariaDB через DB Client
54320	Magnus PostgreSQL порт	Подключение к PostgreSQL через DB Client
63790	Magnus Redis порт	Подключение к Redis через DB Client
30000-30100	Magnus Oracle порты	Подключение к Oracle через DB Client, диапазон портов может быть настроен

Настройка HTTPS и обратного прокси для веб-интерфейса JumpServer

Для чего нужен обратный прокси JumpServer?

Nginx отвечает на поддержку защищенных websockets (wss://), организовывая управление подключениями и защитой канала с помощью SSL-сертификата. Для того чтобы функция копирования и вставки в протоколе RDP работала, необходимо развернуть доверенный SSL-сертификат. Использование копирования и вставки в RDP-активах возможно при доступе через протокол HTTPS.

Установка SSL сертификата и настройка HTTPS для веб-интерфейса

Подготовьте SSL-сертификат (обратите внимание, что сертификат **должен быть в формате PEM**). Сертификаты необходимо разместить в директории **/opt/jumpserver/config/nginx/cert**

Остановите сервис JumpServer:

```
./jmsctl.sh stop
```

Откройте файл конфигурации JumpServer

```
vi /opt/jumpserver/config/config.txt
```

Найдите и измените параметры для конфигурации Nginx:

```
## Конфигурация Nginx
HTTP_PORT=80
SSH_PORT=2222
RDP_PORT=3389

## HTTPS Конфигурация
HTTPS_PORT=443              # Внешний порт для HTTPS, по умолчанию 443
SERVER_NAME=www.domain.com   # Ваш домен для HTTPS
SSL_CERTIFICATE=xxx.pem      # Имя вашего сертификата в /opt/jumpserver/config/nginx/cert
SSL_CERTIFICATE_KEY=xxx.key  # Имя файла ключа в /opt/jumpserver/config/nginx/cert
```

Сохраните изменения файла конфигурации и запустите JumpServer

```
./jmsctl.sh start
```

Если вам нужно дополнительно настроить файл конфигурации Nginx:

```
vi /opt/jumpserver/config/nginx/lb_http_server.conf
```

Многоуровневый обратный прокси на Nginx

Подсказка:

Эта конфигурация подходит для случаев, когда на верхнем уровне есть общий внешний прокси-сервер. Это пример многоуровневого обратного проксирования на Nginx. Каждая прокси-секция должна быть настроена для поддержки длительных соединений WebSocket.

Редактирование конфигурационного файла:

```
vi /etc/nginx/conf.d/jumpserver.conf
```

Пример конфигурации без SSL:

```
server {

    listen 80;
    server_name demo.jumpserver.org;  # Замените на ваш домен

    client_max_body_size 4096m;  # Ограничение на максимальный размер загружаемых файлов

    location / {
        # Здесь указывается IP-адрес Nginx сервера JumpServer
        proxy_pass http://192.168.244.144;
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_request_buffering off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
    }
}
```

Рекомендация:

Для более безопасного доступа рекомендуется настроить SSL и использовать протокол HTTPS, следуя рекомендациям Mozilla SSL Configuration Generator.

Пример конфигурации с SSL:

Перенаправление HTTP на HTTPS:

```
server {
    listen 80;
    server_name demo.jumpserver.org;  # Замените на ваш домен
    return 301 https://$server_name$request_uri;  # Перенаправление всех HTTP-запросов на HTTPS
}
```

Настройка HTTPS:

```
server {
    listen 443 ssl http2;
    server_name demo.jumpserver.org;  # Замените на ваш домен
    ssl_certificate sslkey/1_jumpserver.org_bundle.crt;  # Укажите путь к вашему SSL-сертификату
    ssl_certificate_key sslkey/2_jumpserver.org_bundle.key;  # Укажите путь к ключевому файлу сертификата
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
    ssl_protocols TLSv1.1 TLSv1.2;
    add_header Strict-Transport-Security "max-age=63072000" always;

    client_max_body_size 4096m;  # Ограничение на размер загружаемых файлов и записей

    location / {
        # Здесь указывается IP-адрес Nginx сервера JumpServer
        proxy_pass http://192.168.244.144;
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_request_buffering off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
    }
}
```

3. Другие балансировщики нагрузки (SLB)

Подсказка:

1. Необходимо правильно настроить поддержку длительных соединений WebSocket.
2. Важно учитывать вопросы, связанные с управлением сессиями.

Настройка HAProxy для HA-кластера JumpServer

HAProxy (High Availability Proxy) — это **программный инструмент с открытым исходным кодом** для балансировки нагрузки и проксирования трафика на уровне сетевого протокола, обычно используемый для распределения нагрузки между несколькими серверами. Он является одним из самых популярных решений для повышения доступности и производительности веб-приложений и служб.

Установка HAProxy, на примере Ubuntu:

```
sudo apt install haproxy -y
```

После установки нужно отредактировать файл конфигурации, это вызывает основные вопросы по настройке.
Файл конфигурации обычно находится в **/etc/haproxy/haproxy.cfg**

Пример файла конфигурации с сайта вендора:

```
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats
```

```
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------

defaults
    log                     global
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

listen stats
    bind *:8080
    mode http
    stats enable
    stats uri /haproxy                      # Страница мониторинга, измените по необходимости: http://
192.168.100.100:8080/haproxy
    stats refresh 5s
    stats realm haproxy-status
    stats auth admin:KXOeyNgDeTdpeu9q       # # Учетная запись и пароль для доступа к http://
192.168.100.100:8080/haproxy


#---------------------------------------------------------------------
# check  Параметры проверки активности
# inter  Интервал времени, единица: миллисекунды
# rise   Количество последовательных успешных попыток, единица: раз
# fall   Количество последовательных неудачных попыток, единица: раз
# Пример: inter 2s rise 2 fall 3
# Означает проверку состояния каждые 2 секунды, 2 последовательных успеха указывают на
нормальную работу сервиса, 3 последовательных неудачи указывают на ошибку в сервисе
#
# server Параметры сервера
# server 192.168.100.21 192.168.100.21:80 weight 1 cookie web01
# Первый 192.168.100.21 используется как идентификатор для отображения, его можно
заменить на любой другой строковый идентификатор
# Второй 192.168.100.21:80 — это фактический адрес и порт задействованного сервиса
# weight указывает на вес сервера, при наличии нескольких узлов балансировка нагрузки
будет зависеть от веса
# cookie определяет, что на стороне клиента в cookie будет содержаться этот идентификатор,
чтобы различать текущий используемый задний узел
# Пример: server db01 192.168.100.21:3306 weight 1 cookie db_01
```

```
#---------------------------------------------------------------------

listen jms-web
    bind *:80                           # Прослушивание порта 80
    mode http

    # redirect scheme https if !{ ssl_fc }  # Перенаправление на https
    # bind *:443 ssl crt /opt/ssl.pem       # Настройка https

    option httpchk GET /api/health/         # Интерфейс проверки активности Core

    stick-table type ip size 200k expire 30m
    stick on src

    balance leastconn
    server 192.168.100.21 192.168.100.21:80 weight 1 cookie web01 check inter 2s rise 2 fall 3  #
JumpServer 
    server 192.168.100.22 192.168.100.22:80 weight 1 cookie web02 check inter 2s rise 2 fall 3
    server 192.168.100.23 192.168.100.23:80 weight 1 cookie web03 check inter 2s rise 2 fall 3
    server 192.168.100.24 192.168.100.24:80 weight 1 cookie web03 check inter 2s rise 2 fall 3

listen jms-ssh
    bind *:2222
    mode tcp

    option tcp-check

    fullconn 500
    balance source
    server 192.168.100.21 192.168.100.21:2222 weight 1 check inter 2s rise 2 fall 3 send-proxy
    server 192.168.100.22 192.168.100.22:2222 weight 1 check inter 2s rise 2 fall 3 send-proxy
    server 192.168.100.23 192.168.100.23:2222 weight 1 check inter 2s rise 2 fall 3 send-proxy
    server 192.168.100.24 192.168.100.24:2222 weight 1 check inter 2s rise 2 fall 3 send-proxy

listen jms-koko
    mode http

    option httpclose
    option forwardfor
    option httpchk GET /koko/health/ HTTP/1.1\r\nHost:\ 192.168.100.100  # KoKo , host
HAProxy  ip 

    cookie SERVERID insert indirect
    hash-type consistent
    fullconn 500
    balance leastconn
    server 192.168.100.21 192.168.100.21:80 weight 1 cookie web01 check inter 2s rise 2 fall 3
    server 192.168.100.22 192.168.100.22:80 weight 1 cookie web02 check inter 2s rise 2 fall 3
    server 192.168.100.23 192.168.100.23:80 weight 1 cookie web03 check inter 2s rise 2 fall 3
    server 192.168.100.24 192.168.100.24:80 weight 1 cookie web03 check inter 2s rise 2 fall 3
```

```
listen jms-lion
    mode http

    option httpclose
    option forwardfor
    option httpchk GET /lion/health/ HTTP/1.1\r\nHost:\ 192.168.100.100  # Lion 
HAProxy  ip 

    cookie SERVERID insert indirect
    hash-type consistent
    fullconn 500
    balance leastconn
    server 192.168.100.21 192.168.100.21:80 weight 1 cookie web01 check inter 2s rise 2 fall 3
    server 192.168.100.22 192.168.100.22:80 weight 1 cookie web02 check inter 2s rise 2 fall 3
    server 192.168.100.23 192.168.100.23:80 weight 1 cookie web03 check inter 2s rise 2 fall 3
    server 192.168.100.24 192.168.100.24:80 weight 1 cookie web03 check inter 2s rise 2 fall 3

listen jms-magnus
    bind *:30000
    mode tcp

    option tcp-check

    fullconn 500
    balance source
    server 192.168.100.21 192.168.100.21:30000 weight 1 check inter 2s rise 2 fall 3 send-proxy
    server 192.168.100.22 192.168.100.22:30000 weight 1 check inter 2s rise 2 fall 3 send-proxy
    server 192.168.100.23 192.168.100.23:30000 weight 1 check inter 2s rise 2 fall 3 send-proxy
    server 192.168.100.24 192.168.100.24:30000 weight 1 check inter 2s rise 2 fall 3 send-proxy
```

После изменения файла конфигурации запустите службу haproxy:

```
systemctl enable haproxy
systemctl start haproxy
```

Описание и пример настройки HA-кластера JumpServer

JumpServer (в том числе Community Edition) полностью поддерживает кластеризацию HA без каких-либо ограничений. В этой статье я покажу, как это работает.

Зачем нужен HA-кластер для JumpServer

HA (High Availability) кластер для JumpServer необходим для обеспечения высокой доступности системы и минимизации простоев. Он позволяет:

1. **Избежать простоев**: Если один из узлов кластера выходит из строя, другой продолжает обслуживать запросы пользователей.
2. **Обеспечить отказоустойчивость**: Кластеризация позволяет системе автоматически переключаться на доступные узлы при возникновении ошибок.
3. **Увеличить производительность**: Нагрузка распределяется между несколькими узлами, что улучшает отклик системы при одновременном использовании большого числа пользователей.
4. **Повысить надежность хранения данных**: Использование общих ресурсов, таких как MySQL и Redis, с поддержкой кластеризации минимизирует риск потери данных.
5. **Масштабируемость**: Кластер можно легко расширить, добавив дополнительные узлы для обслуживания большего числа пользователей и задач.

Эта архитектура особенно важна для организаций, где JumpServer используется как критически важная система доступа и контроля.

Компоненты кластера JumpServer.

Ноды\Узлы JumpServer - основные узлы кластера, сервера с установленным JumpServer, каждый сервер не содержит "полезных" данных, можно клонировать\копировать, удалять\добавлять узлы JumpServer.

База данных MySQL\PostreSQL - основная СУБД для хранения всех данных JumpServer: хранит настройки системы, параметры устройств, учетных записей, пароли к целевым системам. Также по умолчанию хранит текстовые логи сессий: команды SSH, SQL запросы, введенные команды с клавиатуры в RDP сессии.
По умолчанию JumpServer создает\использует PostreSQL внутри контейнера на том же сервере, где устанавливаете JumpServer.

База данных Redis - вспомогательная база данных для кэширования, может быть как единой для всего кластера так и отдельными базами для каждой ноды кластера.

По умолчанию JumpServer создает\использует Redis внутри контейнера на том же сервере, где устанавливаете JumpServer.

Хранилище видеозаписей - по умолчанию хранит записи сессий в папке с продуктом **$folder/core/data/media,** где **$folder** - папка указанная в основном конфиг-файле, по умолчанию **VOLUME_DIR=/data/jumpserver**. Через веб-интерфейс продукта можно настроить внешнее хранилище видеозаписей: **SFTP, S3, Ceph, MinIo и другие**

Хранилище логов команд - по умолчанию хранятся в основной БД, через веб интерфейс можно настроить хранение команд и запросов в **Elasticssearch**.

Балансировщик - обычно на базе HAProxy, но можно использовать любой.

Архитектура кластера.

Обычно кластер JumpServer состоит из 2 и более узлов кластера, которые:

- подключены к общей БД(или кластеру) MySQL\PostreSQL
- подключены к общему Redis(или каждый к своему)
- имеют общую СХД для хранения видеозаписей:
  - общая папка данными **$folder/core/data/ (реализуется обычно с помощью NFS сервера)**
  ИЛИ
  - общая внешняя СХД для хранения сессий: SFTP, S3, Ceph, MinIo и другие
  ИЛИ
  - отдельная СХД для каждой ноды
- имеют общую СХД для записей логов команд:
  - на базе общей БД(по умолчанию)
  ИЛИ
  - на базе общей внешней Elasticsrarch
  ИЛИ
  - отдельный Elasticsearch для каждой ноды кластера
- балансировщик

Пример создания кластера HA JumpServer из 2 узлов

Пример создания кластера JumpServer
- с единой общей БД MySQL
- с единой общей БД Redis
- с общей папкой для хранения видеозаписей **$folder/core/data/** с помощью NFS сервера Linux

Для этого нам потребуется:

1. **Сервер с NFS, MySQL, Redis**:
   - 4 CPU, 8 ГБ оперативной памяти.
2. **Узел JumpServer Node1**:
   - 4 CPU, 8 ГБ оперативной памяти, 100Гб свободного места на диске
3. **Узел JumpServer Node2**:
   - 4 CPU, 8 ГБ оперативной памяти, 100Гб свободного места на диске
4. **Сервер HAProxy** (или другой балансировщик нагрузки).

1. Подготовка сервера с NFS, MySQL и Redis

- **Сервер**: Ubuntu 22.04, IP: `10.10.50.10`

Установка и настройка NFS

Команды могут отличаться для других версий Linux, но в целом нужно создать общую папку:

```
sudo apt install nfs-kernel-server
sudo mkdir -p /data
sudo chown -R nobody:nogroup /data/
sudo chmod 777 /data/
sudo nano /etc/exports
```

Добавьте строку в файл `/etc/exports`:

```
/data 10.10.50.10/24(rw,sync,no_subtree_check)
```

Примените настройки и перезапустите NFS-сервис:

```
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

Установка и настройка MySQL

Инструкции зависят от версии ОС. Для создания базы данных и пользователя выполните:

```
mysql -uroot
mysql> create database jumpserver default charset 'utf8';
mysql> set global validate_password_policy=LOW;
mysql> create user 'jumpserver'@'%' identified by 'KXOeyNgDeTdpeu9q';
mysql> grant all on jumpserver.* to 'jumpserver'@'%';
mysql> flush privileges;
mysql> exit;
```

Не забудьте настроить файрволл для открытия порта MySQL (`3306`).

Установка и настройка Redis

Инструкции зависят от версии ОС. После установки Redis выполните:

```
sed -i "s/bind 127.0.0.1/bind 0.0.0.0/g" /etc/redis.conf
sed -i "561i maxmemory-policy allkeys-lru" /etc/redis.conf
sed -i "481i requirepass KXOeyNgDeTdpeu9q" /etc/redis.conf
```

Это позволит доступ к Redis с паролем `KXOeyNgDeTdpeu9q`. Обязательно используйте уникальный пароль для вашего сервера. Откройте порт `6379` в файрволле.

2. Установка JumpServer

Установка первой ноды JumpServer

Монтирование каталога NFS

Установите клиент NFS, смонтируйте папку и настройте автоматическое монтирование при загрузке:

```
sudo apt install nfs-common
mkdir -p /opt/jumpserver/core/data
mount -t nfs 10.10.50.10:/data /opt/jumpserver/core/data
echo "10.10.50.10:/data /opt/jumpserver/core/data nfs defaults 0 0" >> /etc/fstab
```

Настройка конфигурации JumpServer

Редактируйте файл `config-example.txt` в папке установщика:

```
# Измените следующие параметры, остальные оставьте по умолчанию.
# ВАЖНО: SECRET_KEY должен совпадать на всех узлах JumpServer, иначе данные не будут расшифровываться.

VOLUME_DIR=/opt/jumpserver

SECRET_KEY=
BOOTSTRAP_TOKEN=
LOG_LEVEL=ERROR
SESSION_EXPIRE_AT_BROWSER_CLOSE=True

# MySQL

DB_HOST=10.10.50.10
DB_PORT=3306
DB_USER=jumpserver
DB_PASSWORD=KXOeyNgDeTdpeu9q
DB_NAME=jumpserver

# Redis

REDIS_HOST=10.10.50.10
REDIS_PORT=6379
REDIS_PASSWORD=KXOeyNgDeTdpeu9q

# KoKo Lion
SHARE_ROOM_TYPE=redis
REUSE_CONNECTION=False
```

Запустите установку:

```
./jmsctl.sh install
```

После завершения установки вы получите значения для:

```
SECRET_KEY=kWQdmdCQKjaWlHYpPhkNQDkfaRulM6YnHctsHLlSPs8287o2kW
BOOTSTRAP_TOKEN=KXOeyNgDeTdpeu9q
```

Установка второй ноды JumpServer

Установите клиент NFS, смонтируйте папку точно также как на первой ноде.

При редактировании файла конфигурации JumpServer **заполните значения** `SECRET_KEY` **и** `BOOTSTRAP_TOKEN` , полученные после установки первой ноды:

```
VOLUME_DIR=/opt/jumpserver

SECRET_KEY=kWQdmdCQKjaWlHYpPhkNQDkfaRulM6YnHctsHLlSPs8287o2kW
BOOTSTRAP_TOKEN=KXOeyNgDeTdpeu9q
LOG_LEVEL=ERROR
SESSION_EXPIRE_AT_BROWSER_CLOSE=True

# MySQL

DB_HOST=10.10.50.10
DB_PORT=3306
DB_USER=jumpserver
DB_PASSWORD=KXOeyNgDeTdpeu9q
DB_NAME=jumpserver

# Redis

REDIS_HOST=10.10.50.10
REDIS_PORT=6379
REDIS_PASSWORD=KXOeyNgDeTdpeu9q

# KoKo Lion
SHARE_ROOM_TYPE=redis
REUSE_CONNECTION=False
```

И запустите установку

```
./jmsctl.sh install
```

Результат

После завершения настройки вы получите два узла JumpServer, которые используют один MySQL/Redis сервер и хранилище NFS. Вы можете использовать любой из узлов для доступа к целевым устройствам или настроить HAProxy для автоматического перенаправления пользователей на активный узел.